Videogame and Entertainment Industry Standard Sound Design Techniques and Architectures for Use in Videogames, Virtual Environments and Training Systems

Russell Shilling, Ph.D.
Eric Krebs
MOVES Institute
833 Dyer Road, Room 254
Naval Postgraduate School
Monterey, CA 93943-5118
Phone: (831) 656-2543
Fax: (831)656-2595

rdshilli@nps.navy.mil

Keywords: audio, OpenAl, DirectSound3D, virtual environments, sound design, audio architectures, Spatialized sound, videogames, entertainment, America's Army, VoIP, live voice

Videogame and Entertainment Industry Standard Sound Design Techniques and Architectures for Use in Videogames, Virtual Environments and Training Systems

Russell Shilling, Ph.D.
Eric Krebs
MOVES Institute
833 Dyer Road, Room 254
Naval Postgraduate School
Monterey, CA 93943-5118
(831) 656-2543
rdshilli@nps.navy.mil

ABSTRACT: *The design and development of auditory interfaces in virtual environments has lagged behind visual interfaces. However, the auditory interface should be considered an essential component to VE that adds ambience, emotion, and a sense of presence to the simulation. The entertainment industry has long recognized the importance of sound to add these perceptual elements to film and videogames. As a result, the film industry has devoted many of its resources to developing techniques for producing sound effects and ambiences that evoke emotional responses and immerse the viewer in the film. Many of these techniques can be applied to simulations and virtual environments. The current paper will discuss how auditory researchers at the MOVES Institute have applied these sound design techniques used by the entertainment industry for creating highly immersive environments in both their videogame efforts and in high-end simulations. In order to implement this type of sound design, the audio architecture must be capable of producing a complex, spatialized auditory environments. Currently, videogame development tools usually offer more flexibility and control in developing auditory environments than tools used for creating virtual environments. Thus, we look at the use of OpenAL, DirectSound, DirectSound3D, and EAX as alternatives to more expensive audio servers. Additionally, on the high-end, we examine Ausim3D's Goldserver Audio System for employment of low-latency live voice in virtual environments.*

1. Introduction

The MOVES Institute in Monterey, CA has the unique distinction of being a center for high-end simulation as well as professional videogame production. In May, the MOVES Institute in conjunction with the U.S. Army announced the videogame, "America's Army: Operations." America¹s Army: Operations nicely demonstrates the MOVES Institute's prowess in combining entertainment technologies with research and development for defense. Veteran artists, designers, and programmers were recruited from the videogame industry. On the research side, graduate students from all branches of the U.S. military and a number of allied countries worked with MOVES faculty to formulate the theory that propels America¹s Army into new gaming regions. Practical support from the Army included unprecedented access to posts, equipment, and subject-matter experts. For the audio design portion, we turned to the entertainment industry for advice. In the process, our software has been EAX and Dolby Digital certified. Meanwhile these same techniques are being incorporated in our own in-house developed software efforts for creating high-end simulations.

The entertainment industry has long recognized the importance of properly designing sound effects and sound systems to add realism, emotion, and a sense of immersion to film and to video games. As a result, the film industry has allocated significant resources to developing techniques for the design of sound effects and ambient sounds that evoke a sense of realism and manipulate the emotional response of the viewer. It is difficult to imagine that all sound heard in the battle scenes of "Saving Private Ryan" were added in layers after the film was shot, yet they were. In the opening scenes depicting the Normandy invasion, the audio effects, including the actor's voices, are completely synthetic; added to the film after it was shot. The audio effects were spatialized using a surround sound system to immerse the audience in the sound field.

Even though the medium is non-interactive, a movie with a properly designed audio track represents a better example of an auditory virtual environment than most high-end simulations developed by industry and the military. Although film is a non-interactive medium (for most people), videogames are another matter. These same audio design techniques have been adapted by the electronic gaming industry to add both realism and excitement to their titles. Although, not reported here, current research in our laboratory is aimed at deriving objective rather than subjective measures for the contributions these techniques add to virtual environments using a variety of techniques including psychophysical measurement, electrophysiology and even basic memory experiments [9]. These measurement techniques are leading to new methods for assessing immersion in virtual environments.

There are many lessons to be learned from the entertainment industry. First is in the field of sound design. One theme has emerged from discussions with professional sound designers, producers, and audio engineers. Sound is emotion! Anyone in the audio industry will tell you this. A visual display lacking a properly designed audio component will be emotionally flat. Sound design for interactive environments is not a trivial matter in either its importance or implementation. As will be discussed, sound design needs to be a carefully planned process.

The second lesson to be learned is in the way we can implement an enriched audio environment in an interactive simulation. Traditionally, spatial audio in high-end simulation has been a very expensive proposition. Audio servers can run from $20k to well over $50k, putting them out of the reach of many projects. Our laboratory has been comparing the performance of these high-end systems with the performance offered by off-the-shelf and open source technologies. The second part of this paper will explore these "low-end", "low-cost" architectures for presenting Spatialized sound while still creating a highly immersive auditory experience.

2. Sound Design

The first rule of sound design is, "see a sound, hear a sound." [3] Unless we supply the appropriate background sounds (machinery, artillery, animals, footsteps, etc) the participant will probably feel detached from the action. Footsteps sound differently when you're in the grass as opposed to the pavement or in a hallway. Likewise, the sound of the action of an M-16 changes depending on whether you're located inside a building or outside. Explosions and other concussive events are not just auditory; they are tactile, full-body experiences. Vehicle noise is very specific to a particular vehicle and becomes part of the expectation for using that device. These are the types of things that create immersion, ambience and emotion in movies, and the same should hold true in virtual environments. Four of the audio elements we've been focusing on in our development of virtual environments and videogame based simulations are footsteps, vehicular sounds, weapons, and general ambience.

2.1 Footsteps

As we are discovering in our military videogame development activities, these techniques should be applicable to the types of virtual environments and simulations currently under development by military establishments. The entertainment industry employs foley actors to create footsteps for films. Footsteps are a rather small detail that sound designers believe (an unanswered empirical question) adds to the sense of presence in a film, even in conditions where footsteps may not actually be heard in the real world. Current game editors allow us to create the sound of footsteps on different terrains and textures. This should be a *de facto* standard for all VE development for dismounted infantry. In fact, it has been our experience that the implementation of audio in the editors of gaming engines are far more advanced than in so-called, "high-end simulations".

2.2 Vehicular Sounds

Likewise, vehicle noises should be as accurate and as detailed as possible. Recordings and sound levels should be available for sounds inside, behind, and in front of the vehicles [3]. For instance, for various military amphibious vehicles, the sounds of wheels on different terrain surfaces may be very important for creating a sense of presence. Details such as engine idle, engine revs, and gear changes may need to be included, both in the water and on land. Our laboratory has recently applied professional recording techniques for creating audio for an LCAC using professional sound designers and recording engineers to excellent results. Reproduced over a high quality audio system, the result is very accurate and very compelling. Other platforms will soon be following suit.

2.3 Weapons Sounds

Weapon sounds provide a particularly difficult challenge for simulations designers. One of the first things you will hear from customers of military simulations is that they want the weapons to sound accurate. Practically speaking, this is impossible. Although the mechanisms and actions of weapons (foley) can be accurately recorded and replicated, the actual sound of a weapon firing is very problematic. First, microphones cannot capture the full dynamic range of the majority of concussive weapons. Likewise, commercial speaker and headphone systems cannot produce this dynamic range. Therefore, an M16 will never sound exactly like an M16. From discussions with various videogame companies and sound designers for films, attempts to use accurate recordings of weapons result in an uninteresting and emotionless perception of the auditory event. Listeners complain that they don't sound realistic. There are several solutions to this problem. One solution is to record weapons with multiple tape recorders using different types of microphones that accentuate different parts of the frequency spectrum, then mix the recordings together [3]. This tends to be expensive and tedious since you need two sets of expensive recording gear. For smaller budgets, the solution is to dive into your sound effects libraries and look for weapons sounds that combine the audio attributes you desire, then mix them together [3]. These techniques produce effects that are "hyper-real" [1]. The sound of an actual M-16 firing may not sound as exciting as that of an M-16 mixed with other weapons to produce something more compelling. Remember, the best you can accomplish is producing a gunshot that sounds SIMILAR to the actual weapon. The worst thing you can do is simply use your editing software to equalize the audio and boost the bass response. A large part of the emotional response is due to the high frequency crack of the weapon and not, as many people believe, the low frequency boom.

2.4 General Ambience

The audio element which is probably the most important aspect of evoking the sense of immersion in a VE is ambient sound. It's also the least likely to be implemented. The sound of a breeze, a computer fan, an air conditioner, or an electrical generator may not be obvious to the main training focus of a simulation, but they will contribute to the sense of presence or immersion. It may seem like a trivial point at first, but we do not live in a silent world. We create surrealistic unreality in VE by failing to include these basic cues. We don't notice them when they are there. But, we do notice their absence. One of the things we are currently attempting to demonstrate is that these ambient sounds do, in fact, impact performance and training in VE.

3. Sound Libraries vs Custom Recording

There are some very fine prepackaged professional audio libraries. Good libraries are not cheap, but worth the expenditure. We've found that sound libraries provide excellent weapons foley, footsteps, and a wide variety of vehicles. They do not typically do a fantastic job for explosions and weapon firing. The sound designer will have to do significant amounts of work to get these sounds to have a significant impact on the listener. Not surprisingly they also lack the more esoteric weapons platforms and vehicles. Hence our recent foray to record an LCAC. If you do not have your needed sound in a library, you will either have to synthesize your sounds from recordings of similar equipment, or go out and record it yourself.

Detailed instructions for obtaining these recordings are readily available from experts and publications on the topic of "sound in film" [1,2]. However, be warned that recording audio is as much of an art as a science. Digital recording is not always better, especially when recording explosions. There is a dizzying array of microphones, each with different frequency response and directionality. Just because you've gone out and bought a $1500 digital tape recorder, does not mean you're ready to start recording effects. So, be advised that recording equipment can be very expensive, and you're going to have to learn to use it properly. Our laboratory is currently using a Tascam DA/P1, which is an excellent quality digital recorder commonly used as a field recorder and is capable of handling professional quality microphones. We will be purchasing (or leasing) an analog recorder in the near future to handle the weapons sounds mentioned earlier.

If you don't want to spend large amounts of time learning the art of recording and audio postproduction and you have a limited budget, it might be better and cheaper to hire a professional to do the work for you. The bottom line is that sound design is a very complex issue and deserves far more attention in the development of virtual environments and training systems than it is currently awarded.

4. Videogame Audio Architectures

When creating immersive audio for videogames, there are several techniques (or combinations) that the designer and programmers has to create spatial sound. Although these technologies are still much cruder than what we need to produce and extremely realistic auditory environment, most videogame development platforms give the designer far more options than the average virtual environments development platform.

4.1 OpenAL

OpenAL is a cross-platform audio API designed to provide a software developer with a simple interface to a spatialized audio capability. It is in the public domain and is open source code. The primary force behind the development of OpenAL is Creative Labs, who use it as the primary audio programming interface for Win32, UNIX, and Linux platforms (Creative does not currently support Mac OS X and provides only minimal support for Mac OS 9 and earlier).

OpenAL is a platform-independent "wrapper" API for whatever platform it is being implemented on. For example, on a Wwin32 platform, OpenAL accesses the DirectSound or DirectSound3D driver. Platform discovery is automatic in OpenAL.

A few of OpenAL's capabilities include:
a. Audio Contexts - a context in OpenAL can best be described as an audio "situation" - an environment consisting of a listener and sounds. OpenAL, like all audio programming API's only support one context per machine, except in those circumstances where a single computer contains multiple sound cards. In those cases, multiple contexts can be implemented for each sound card

b. Spatialized Audio - OpenAL supports one listener per context, and as many sounds as the host machine memory will support. OpenAL will generally manage whether sounds are processed on the sound card (in hardware) or on the host CPU (in software) automatically. The priority is to utilize hardware assets (buffers) first, followed by software. Most sound cards support between 16 and 64 hardware-processed, simultaneously playing sounds. Spatialization of audio is accomplished by constructing buffers of sound data - OpenAL currently supports multiple audio formats, including PCM wave files and MP3 formatted data. The data is loaded into memory through a simple API call, and the API returns a handle to the data for subsequent playing or looping. The developer has control over the how the sound file is played; playing looping, stopping and restarting, rewinding are all inherent capabilities. Spatialization is accomplished through a simple positioning method that is source-specific for each buffer. The single listener can also be position independently of each sound source. One aspect of OpenAL that is different from other audio API's is the separation of the audio source from the audio data. A source is a buffer that can be positioned - it is not tied directly to any single wave file or sound - the source can be positioned and any wave file can be played through that source. While this may complicate programming for a beginning audio programmer, this distinction is very powerful.

c. Audio Rolloff and Attenuation - OpenAL provides for audio rolloff (attenuation of audio sources based on distance from the listener) through three different attenuation models - inverse distance, inverse distance clamped, and exponential. Selection of rolloff model is left to the developer. Manual attenuation of sources can be accomplished through volume settings specific to each source.

d. Static and Streaming Audio - OpenAL supports both static buffers (buffers whose data is loaded completely and stored in memory) and streaming buffers (buffers that contain only a portion of the audio data at creation, but continually read new chunks of data as specified intervals). Streaming audio permits the application developer to play extremely large wave files without the memory of CPU penalty of storage and retrieval. Control over the "feeding" of audio data is exposed to the developer.

e. Pitch Bending or Frequency Shifting - OpenAL supports frequency modification of audio sources at execution time. This is especially beneficial when modeling sounds such as automotive engines - the pitch of the engine sound can be modified to reflect change in velocity.

f. Doppler Processing - OpenAL supports audio source Doppler effects. OpenAL does not calculate audio source velocity and automatically modify source frequencies for the Doppler effect, but manual setting of velocity parameters will permit Doppler effect processing. One shortcoming to be mentioned here is that, in its current release, OpenAL does not permit setting of a reference velocity for Doppler calculations. OpenAL has "hard-coded" the speed of sound at sea level (in meters/sec) as its reference velocity. This restrains the developer from being able to amplify the Doppler effect. While from a physically-based modeling perspective a developer may not be inclined to change the physical properties of a sound, developing audio software from an entertainment perspective may require certain audio effects to amplified or diminished. OpenAL does not support this. A typical example is that of a train approaching and departing a listener's position.

Combining all of the above capabilities into a single, platform-independent API makes OpenAL extremely useful. OpenAL is implemented in many PC games, including "America's Army: Operations" developed at the Naval Postgraduate School which uses Epic's Unreal Warfare engine. One significant limitation of OpenAL as an audio API is that it does not directly support live voice audio. Live voice is a critical component of many applications, such as training simulators or PC games involving multiple participants distributed across a network. Live voice audio can be accomplished with OpenAL, but it would require the use of 3rd -party API's for voice capture and encoding, network transmission of voice data, and decoding. Once decoded, the voice data could be used as an input to an OpenAL streaming buffer and position accordingly to provide a spatialized voice effect.

4.2 DirectSound, DirectSound3D and DirectMusic

DirectSound and DirectSound3D are audio programming API's produced by Microsoft. Originally released as individual API's, they are now integrated and released as a core component of Microsoft's DirectX programming suite, currently in release version 8.1. DirectMusic, released for the first time in DirectX 8.0, is a new audio programming API that both wraps DirectSound and introduces several new functionalities.

Instead of re-hashing capabilities of DirectSound that are similar to OpenAL, the following will point of differences with OpenAL and additional capabilities provided by DirectSound:

a. Limited to Wave Files - DirectSound only supports wave file PCM data in its current release. While wave files are the audio programming industry's format of choice, this limitation excludes using other types of sound data, and may require programmers to obtain additional software capable of converting audio files to PCM format.

b. Integration of Data and Sources - DirectSound tightly integrates sound data and the buffer through which it will be played. A static buffer can only be loaded with audio data once. It can be played and repositioned, as many times as the developer desires, but it can never accept new data. In OpenAL, different audio data could be played though the same buffer. For streaming buffers, functionality of OpenAL and DirectSound is equivalent - audio data can be continually fed into a streaming buffer and replaced.

c. Audio Effects - DirectSound supports seven different types of audio effects processing directly within its API - echo, gargle, compressor, chorus, distortion,flanger, and a limited reverb. While these effects are relatively simple and not extremely flexible (such as those found in EAX - see below), they do provide the programmer with access to a limited range of audio effects without having to import another API; examples include room echo and room reverberation

d. Live Voice - perhaps the single greatest advantage to DirectSound over any other API is its integration of live voice. For those applications requiring live voice, DirectX contains another core module, called DirectPlay, which supports networking on a client-server or peer-to-peer structure, which contains a sub-module named DirectVoice. DirectVoice integrates DirectPlay's networking with DirectSound's 3D spatialized audio capability to provide a spatialized live voice capability to any programmer. Currently, DirectVoice will support up to 64 live voices in a session. An example of this would be multiple participants in a virtual world communicating with their voice positions concurrent with their avatar positions.

The other capabilities of OpenAL are similar to those found in DirectSound. DirectMusic is an audio programming API designed to support musicians more than application developers, but has added capabilities that game or simulation developers may benefit from using. Some of these include:

a. Multiple Audio Formats - DirectMusic supports multiple audio file formats, including MP3, wave an others.

b. DLS (Downloadable Sounds) - DLS is a standard for integrating several audio files for synchronized playback. While both OpenAL and DirectSound can shift frequencies, audio artifacts and distortions when the shift is far away from the original frequency will become apparent. DLS permits a programmer to use, for example, three audio files - one for low frequency sounds, one for intermediate frequency sounds and one for high frequency sounds, and then overlay them as necessary during playback. This precludes high- and low-pitch shifted artifacts and can produce much more realistic sounds in this type of scenario.
Modeling vehicle engines provides an excellent example of the power of this feature. Three different audio files could represent low RPM, intermediate RPM, and high RPM. Synchronized overlay and playback would preclude frequency-shifting artifacts and produce a smooth sounding engine.

c. Separation of Buffers from Audio data - DirectMusic uses the OpenAL model of separating a source from its data. In DirectMusic, a source (AudioPath) can be positioned, repositioned and manipulated in whatever fashion the developer desires. Audio data is then placed on the AudioPath when it is time to play.

d. Audio Scripting - DirectMusic supports scripting long segments of sounds and permits the developer to introduce variability. This capability might enhance training simulations where ambient sound tracks could be developed and varied from run to run, without have to re-program audio sequences between each execution. Like OpenAL, DirectX (either DirectSound or DirectMusic) is a full-feature audio programming API capable of delivering an exciting audio experience to the user.

4.3 EAX

EAX Audio Extensions is an audio API produced by Creative Labs to induce numerous types of audio effects, including reverberation, occlusion, obstruction, and exclusion. The goal of the API is to produce effects equivalent to modeling the acoustics of rooms, buildings, and other audio environments. It does this without the expensive CPU requirements of actually modeling geometry and audio ray tracing. EAX 2.0 is the current release version, and 3.0 is close to public release. EAX works as an extension to an underlying audio API - EAX is currently configured to work with both OpenAL and DirectX.

A few of EAX Audio Extension's capabilities include:

a. Audio Environments - EAX 2.0 supports both global and source-specific audio effects. Global effects are applied to the single listener in the environment, while source specific effects are independently applied to audio sources. EAX 2.0 permits the application developer to select from one of 26 defined preset environments for global effects, and five high level parameters to modify those presets to obtain the type of effect the developer desires. Without going into specific parameters that can be modified, EAX supports, though its methods and variables, modifying parameters to:

1. Set or modify room size
2. Set or modify room width or depth
3. Set or modify room materials - for example, a hardwood floor will reflect a greater amount of sound energy than a carpeted floor

4. Set or modify room height

Each of these effects may require multiple parameter changes or method calls, but each is obtainable.

b. Audio Source Effects - as audio sources are moved in the environment, EAX can construct three main types of effects to simulate how those sources would interact with a listener:

1. Occlusion - occlusion is the effect of a listener and an audio source being in different rooms. Depending on the type of material separating the listener from the source (which is fully modifiable in EAX), a variable amount of sound energy will penetrate the separator and arrive at the listener. EAX permits setting of various types of wall or room material.
2. Exclusion - exclusion is the effect a listener hears when the audio source is not in the same environment but is heard through a portal, such as a door or a window. Exclusion is an attenuation of reflected and reverberated sound energy while the direct path energy is relatively unaffected. EAX supports modification of parameters to achieve different effects for door or window size.
3. Obstruction - obstruction refers to the effect where the listener and the sound source are in the same environment, but an object is between the two. In this case, direct-path sound energy is attenuated, especially at high frequencies, and reflected and reverberated sound is left untreated.

All of these three effects have multiple parameters that must be set to achieve the desired overall effect.

EAX can be implemented in any audio application. We would recommend the use of EAX's Unified Interface to most program developers - the Unified Interface provides a single API for all versions of EAX, and even will disable the effects if the host machine's sound card does not support EAX without terminating the application.

The most important issue to understand when using EAX is that it does not employ true "room acoustics". It is strictly a numerical, parameter-based API for achieving certain effects. Experience suggests that numerous iterations of parameter tweaking are required to achieve integrated and synchronous effects for an interactive application where the listener and sources are in constant movement.

In conclusion, both OpenAL and DirectX are extremely powerful audio API's capable of delivering a rich audio experience. When combined with EAX, either can come quite close to simulating an audio experience that rivals reality. DirectX module API's are much more difficult to understand and comprehend, and result in a very steep learning curve for beginning programmers.

Our recommendation is that for beginning programmers, OpenAL provides an easy to understand API that can familiarize one with the basic of audio programming yet still create a dynamic audio environment. If the application requires live voice, DirectX's DirectSound, DirectPlay, and DirectVoice offer the only integrated API available.

5. Ausim3D Goldserve

For multi-participant virtual environments, especially those involving teams or team training, the auditory environment must at least address how players will communicate with each other. The most natural form of communication amongst team members is voice communication, just as it would be in the real world. Adding live voice to virtual environments is expensive and problematic. The biggest obstacle facing a virtual environment developer who wants to incorporate live voice is latency. The DirectX suite mentioned above supports live, spatialized voice, but it faces significant latency issues, generally on the order of 200 – 500 ms, most of which is inherent in Voice over Internet Protocol (VoIP) applications. Latency in voice communications can be highly detrimental to immersion and the sense of presence.

One solution is the utilization of Ausim3D's Goldserve Gold Series Audio Localizing Server System. The Goldserve is capable spatializing live voice with a measured latency of 5 – 10 ms, virtually imperceptible to the participants. Just as with DirectX and OpenAL, distance attenuation, rolloff, and source directivity are additional capabilities. Under testing and examination at the MOVES Institute, initial experiments involving multiple participants in spatialized, live voice virtual environments indicate participants greatly favor the Ausim3D Goldserve to DirectX and other typical VoIP applications.

Future upgrades to the Ausim3D Goldserve will implement a broad range of room acoustic modeling capabilities. When these upgrades are complete, the Goldserve, combining spatialized, live voice with a run-time room acoustics processing capability, will be the premier audio system for developing fully interactive, multi-participant virtual environment training systems and simulations.

5. References

[1] Holman, T. (1997). Sound for Film and Television. Boston, MA: Focal Press.

[2] Holman, T. (2000). 5.1 Surround Sound Up and Running. Boston, MA.: Focal Press.

[3] Yewdall, D. (1999). Practical Art of Motion Picture Sound. Boston, MA: Focal Press.

[4] Letowski, T., Karsh, R., Vause, N., Shilling, R., Ballas, J., Brungert, D. & McKinley, R. (2001). Human Factors Military Lexicon: Auditory Displays. ARL Technical Report, ARL-TR-2526; APG (MD).

[5] Shilling, R.D., Shinn-Cunningham, B. (2002). Virtual Auditory Displays. Virtual Environments Handbook, Kaye Stanney,. New York, Erlbaum.

[6] Langendijk, E. H., and Bronkhorst, A.W. (2000). Fidelity of three-dimensional-sound reproduction using a virtual auditory display. Journal of the Acoustical Society of America 107(1): 528-537.

[7] THX Certified Training Program (2000). Presentation Materials. June 20-23. San Rafael, CA.

[8] International Telecommunications Union. (1994). Multi-channel Stereophonic Sound System With and Without Accompanying Picture. Recommendation ITU-R BS.775-1. Hendrix, C. and Barfield, W. (1996). The sense of presence in auditory virtual environments. Presence 5(3): 290-301.

[9] Sanders, R., and Scorgie, R. (2002). The Effect of Sound Delivery Methods on the User's Sense of Presence in a Virtual Environment. Thesis Research. Naval Postgraduate School, Monterey, CA.